Création et évaluation d'interfaces visuelles interactives à partir de services web

Romain Vuillemot

Création et évaluation d'interfaces visuelles interactives à partir de services web

Romain Vuillemot

INRIA Saclay
Gif-sur-Yvettes, France
romain.vuillemot@inria.fr

RÉSUMÉ. *Cet article présente une architecture logicielle basée sur les services web et permettant la création et l'évaluation d'applications visuelles interactives. Les services web sont une standardisation d'échange de données dans un système distribué, tel que le web. Ils servent principalement à la publication de données (au moyen d'API) stockées dans des bases de données, mais peuvent également servir au traitement de celles-ci. Nous montrons que leur composition permet la création de représentations visuelles de données, en reconstituant le modèle de référence de celles-ci. Les visualisations ainsi générées peuvent être rendues interactives une fois couplées avec un programme interactif (tel qu'un navigateur web), afin de permettre à l'utilisateur de réaliser des tâches d'exploration et d'analyse visuelle de données. Nous présentons une interface de composition de services et l'application à la visualisation de graphes et de nuages de mots. Enfin nous montrons comment l'usage des logs générés côté serveur permet la représentation et l'évaluation de l'activité de l'utilisateur.*

ABSTRACT. *This paper presents a software architecture, based on web services, that enables the creation and evaluation of interactive visual applications. Web services are a standard for data exchange in a distributed system, such as the web. They are mainly used for data publishing (via API), but can also be used for data processing. We show that web services composition permits the creation of data visualization, by reconstructing the reference model. Generated visualizations can be made interactive once coupled with a program (such as a web browser) to let the user perform visual exploration and data analysis tasks. We also present a service composition interface, and applications to graph and word cloud visualization. Finally, we show how generated server-side logs allow the representation and evaluation of users' activity.*

MOTS-CLÉS : *interfaces visuelles, visualisation d'information, services web, évaluation.*

KEYWORDS: *visual interfaces, information visualization, web services, evaluation.*

DOI:10.3166/RIA.26.429-453

1. Introduction

Une étude d'IBM [1] estime que nous produisons 2,5 Exaoctets de données chaque jour et que 90 % des données existantes ont été produites ces deux dernières années. La plupart de ces données sont des ressources disponibles sur le web ou dans des nuages de données (bases de données géographiquement distribuées). Ces données sont souvent mises à jour de manière continue, et l'analyse elle-même de ces données en produit de nouvelles : logs de moteur de recherche, commentaires, liens sociaux, etc. Des outils toujours plus efficaces sont nécessaires pour les analyser, les comprendre et partager leur interprétation.

Le domaine de la visualisation interactive de données a produit des méthodes et outils permettant de donner du sens à ces données (Fekete *et al.*, 2008) et d'en découvrir de nouvelles. Permettre de mieux les explorer et les comprendre, en un temps le plus court possible, n'est pas tant lié au volume de données qu'à l'espace d'exploration en termes de sélections, filtrages et d'associations possibles à des variables graphiques. En partant de l'hypothèse que l'humain (système cognitif, capacités de perception et d'interaction) et l'ordinateur (écran, clavier) restent relativement stables, la voie à explorer est l'optimisation de la bande passante de l'utilisateur via son canal visuel. Une approche permettant de réduire l'espace à explorer est la combinaison de la visualisation interactive avec des méthodes automatiques d'analyse de données. L'utilisateur reste cependant au contrôle de l'interface, pour prendre la décision sur le résultat final et guider les traitements automatiques. Cette approche est connue sous le domaine de l'analyse visuelle de données (*Visual Data Analytics*) qui a pour objectif de fournir des outils afin de réaliser des tâches exploratoires. Les domaines d'application sont nombreux, et vont de l'intelligence économique, jusqu'à l'analyse de données biologiques. Le livre *Illuminating the path* de (Thomas, Cook, 2005) propose un agenda et identifie les principaux verrous à relever, dont celui que constitue l'évaluation des applications qui est considéré comme un pilier essentiel : *Visual analytics must develop meaningful and effective techniques to evaluate the actual value any specific visual analytic technique may provide*. Il est également nécessaire d'avoir des outils techniques flexibles, afin d'être en mesure de traiter des jeux de données et tâches qui n'ont pas été anticipés ou qui n'était pas connus lors de la conception initiale du système.

Nous nous intéressons aux techniques de création et à l'évaluation d'interfaces visuelles interactives à l'échelle du web, et leur extension au domaine de l'analyse visuelle de données. Pour être en phase avec le domaine du web, nous nous plaçons dans le contexte très précis où il n'est pas possible d'observer l'utilisateur autrement que par les systèmes distants qui hébergent les ressources (données) et leur traitement. La conséquence directe est qu'il n'est pas possible de connaître le profil de l'utilisateur, ni ses intentions, et encore moins sa satisfaction éventuelle comme c'est le cas avec une évaluation utilisateur *directe*. Cette approche écologique a certes ses limites, mais

1. Janvier 2012, http://www-01.ibm.com/software/data/bigdata/

est proche des conditions réelles d'usages actuelles et à venir. Nous introduisons tout d'abord quelques exemples de systèmes visuels interactifs (section 1.1), la problématique de l'évaluation dans ce contexte (section 1.2) et enfin l'approche générale et les objectifs de cet article (section 1.3).

1.1. Exemples d'environnements visuels interactifs

Nous présentons tout d'abord cinq systèmes issus du domaine de l'analyse visuelle de données et des interfaces visuelles interactives à l'échelle du web.

GEOVISTA (Takatsuka, Gahegan, 2002) est un système d'exploration visuelle qui offre des vues multiples coordonnées de données multidimensionnelles (figure 1). Les vues peuvent aussi bien être des nuages de points, que des cartes choroplèthes. L'utilisateur peut sélectionner un ou plusieurs éléments (valeurs, dimension) et cette sélection sera propagée sur les autres vues, tout en gardant le contexte, à savoir les données non sélectionnées (mais rendues floues, par exemple, pour indiquer leur non-sélection).

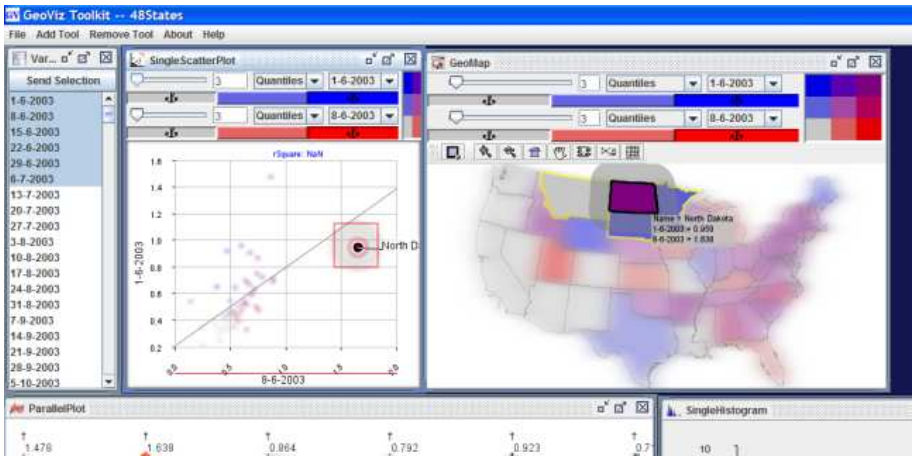

Figure 1. Capture d'écran de l'interface du système GEOVISTA

JIGSAW (Stasko *et al.*, 2008) est un système complet permettant le support de raisonnement analytique au moyen de vues multiples sur les données (figure 2). Le système possède plusieurs mécanismes de visualisation et d'interaction avancés, de vue sur la navigation de l'utilisation, ainsi qu'une particularité supplémentaire de création de vues de connaissance (Shrinivasan, Wijk, 2008). La conséquence de ces différents types de vues, est que plusieurs écrans sont parfois nécessaires pour toutes les afficher simultanément (figure 2).

GAPMINDER (GapMinder, 2010) permet de visualiser automatiquement l'évolution des données graphiques au fil des années (le temps est donc encodé en une variable *physique*, et non visuelle) (figure 3). Des widgets permettent le contrôle du flux

Figure 2. Capture d'écran du système d'analyse visuel JIGSAW

(lecture et pause, ainsi que la vitesse de lecture). L'utilisation de la lecture automatique permet à l'utilisateur de se focaliser sur une tâche d'observation de l'évolution ou de l'explication de cette évolution à une audience.

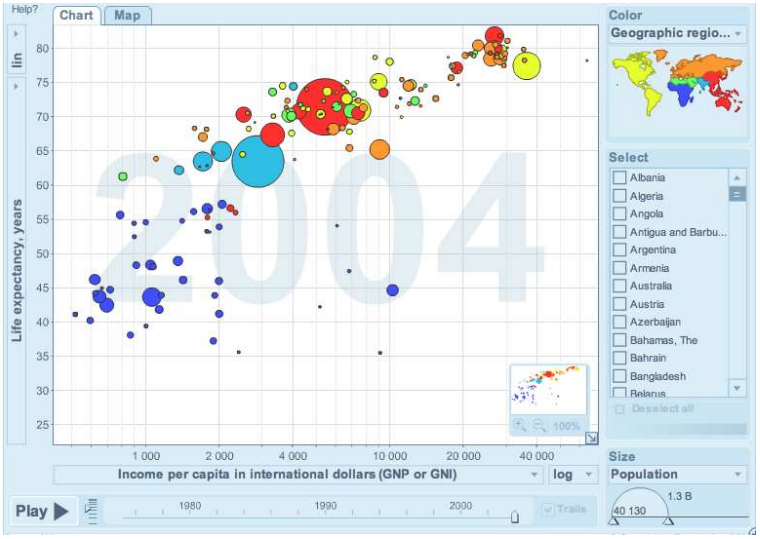

Figure 3. Capture d'écran de l'interface GAPMINDER

IBM MANYEYES (Wattenberg *et al.*, 2007) est un service web (accessible via un site web) permettant la mise en ligne de représentations visuelles, telles que des nuages de mots clés ou des graphes, et leur partage (figure 4). Chaque utilisateur du site doit tout d'abord s'identifier, ensuite télécharger son jeu de données (texte, tables)

et créer de multiples visualisations qu'il personnalise (taille, couleur). Le résultat de la visualisation est disponible sous forme d'URL qui est une vue sur les données et qui peut être partagée, commentée et évaluée par d'autres utilisateurs (paradigme du web 2.0) sur des sites tiers. Cette approche est prometteuse car elle permet désormais une évaluation sociale par rapport à l'approche cognitive classique, si l'usage est massif. Mais le système ne permet ni d'offrir un cadre d'analyse de données sophistiqué, ni d'effectuer des tâches exploratoires car les interactions sont limitées. Enfin, chaque jeu de données téléchargé et visualisation créée sur le site devient publique, ce qui peut limiter l'usage dans le cas de données confidentielles ou sensibles.

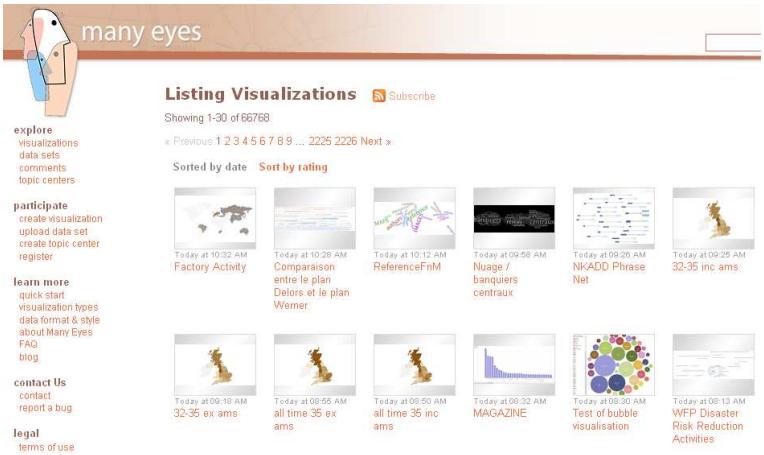

Figure 4. Capture d'écran du site web MANYEYES

GOOGLE INSIGHTS FOR SEARCH (Google Insights for Search, 2010) est un site web permettant l'exploration de tendances (figure 5) et leur distribution géographique et temporelle. Il s'agit d'une combinaison de vues multiples synchronisées avec lesquelles il est possible d'interagir pour obtenir des détails ou les animer. La visualisation globale est initiée par une requête (le ou les mots-clés dont on veut connaître la tendance) qui résulte en une URL unique. Les filtrages locaux, eux, n'apparaissent pas dans l'URL et ne peuvent ainsi pas être partagés.

Parmi ces interfaces visuelles, GEOVISTA et JIGSAW sont exécutées sur la machine de l'utilisateur et proposent des visualisations dont les états (à savoir les vues générées accompagnées de leurs paramètres) restent internes au système. À l'opposé, les interfaces web (GAPMINDER, IBM MANYEYES, GOOGLE INSIGHTS FOR SEARCH) nécessitent la connexion à un système distant, et génèrent des vues identifiables et accessibles au moyen d'une URL souvent unique et diffusable sur le web. Cette dernière propriété est intéressante dans le cas d'analyses visuelles complexes, car elle permet d'accéder, mémoriser, et partager rapidement un résultat. Cependant, ces vues ont une granularité assez élevée et ne reflètent pas des actions comme le filtrage ou l'annotation. Elles ne reflètent pas non plus tout le parcours (hypothèses, erreurs, annotation) qu'a pu effectuer l'utilisateur, car seule la dernière vue est partagée et non pas l'historique dans son ensemble.

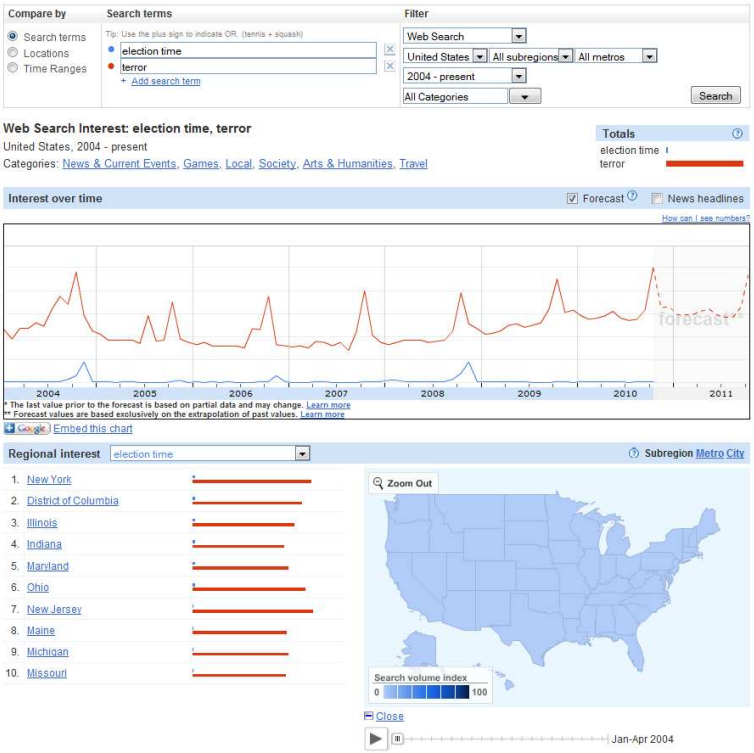

Figure 5. Capture d'écran de l'interface de GOOGLE INSIGHTS FOR SEARCH

1.2. Problématique de la conception d'interfaces visuelles interactives

La conception d'une application visuelle interactive est composée de cycles développement logiciels (en termes de langages et outils) assez classiques, mais a la singularité de nécessiter, pour le programmeur, des compétences sur toute la chaîne de traitement de l'information, du modèle de données jusqu'à la représentation visuelle, en passant par les techniques d'analyse automatique. Ensuite l'application doit minimiser la surcharge visuelle (par exemple le nombre d'éléments présents à l'écran) et cognitive (par exemple le temps d'apprentissage) de l'utilisateur. Enfin l'application étant destinée à résoudre une tâche d'un domaine, il faut préalablement connaître celui-ci et s'adapter aux habitudes de ses utilisateurs.

Ces phases de développement ont déjà fait l'objet de nombreux travaux et de recommandations (Shneiderman *et al.*, 2009 ; Dix *et al.*, 2004 ; Card *et al.*, 1999). Parmi celles-ci (souvent de haut niveau et abstraites comme discuté section 3), certaines ont une validation relativement objective et *quantifiable* telles la *réactivité* et *la boucle de rétroaction* (feedback). La réactivité correspond au temps de réponse du système suite à une action de l'utilisateur. Neilsen (1993) indique que même si ce temps doit être le plus court possible, dans la réalité, cela est rarement le cas, ce qui est pourtant un défi

majeur (Hibbard, 2004). La rétroactivité permet à l'utilisateur de connaître l'état du système suite à une interaction (Norman, 2002). L'utilisateur pourra par exemple savoir si sa requête a été prise en compte, si elle est en cours d'exécution, ou si elle a déjà été exécutée. L'utilisateur peut ainsi éviter de répéter une action et avoir connaissance des actions qu'il peut ou ne peut pas réaliser ensuite.

Malgré tout, même si ces règles (aussi bien de haut que de bas niveau) sont validées, une interface visuelle n'est pas la garantie que l'utilisateur réussisse la tâche qu'il souhaite effectuer. Ces tâches, telles que la recherche d'une information ou la compréhension d'un jeu de données, sont réussies selon l'appréciation de l'utilisateur et non du système. Celui-ci ne peut pas non plus toujours définir le périmètre (autrement dit la session) correspondant à la résolution de la tâche. En effet, les choix que l'utilisateur effectue au fil d'interactions, comme la construction itérative d'une requête, l'exploration des résultats ou les retours en arrière, ont un sens au regard du modèle mental que l'utilisateur se construit de la tâche, qui peuvent être plus ou moins nombreux que ce que le système enregistre. Enfin, un ensemble identique d'interactions et de visualisations peut correspondre à des tâches différentes, et engendrer une évaluation également différente.

1.3. Approche et objectif de l'article

Cet article reprend en partie les travaux de thèse de Vuillemot (2010), et propose une architecture de conception et d'évaluation d'interfaces visuelles interactives à l'échelle du web.

L'architecture que nous proposons est basée sur les services web permettant la création et l'évaluation d'applications visuelles interactives. Les services web sont une standardisation des échanges de données dans un système distribué, tel que le web. Ils servent principalement à la publication de données (au moyen d'API), mais peuvent également servir au traitement de celles-ci. Nous montrons leur application à la création de représentations visuelles de données, par combinaison de services, afin de recomposer le modèle de référence de traitement des données. Cette architecture est en phase avec la structure actuelle du web, pour la publication et l'analyse de données. L'objectif de l'architecture est de faciliter le temps de développement en reprenant ce cadre, qui nous le verrons, permet de couvrir tous les traitements. L'objectif est également d'offrir une flexibilité à l'utilisateur afin d'adapter la visualisation à la tâche qu'il souhaite effectuer, au-delà de celle initialement définie par le programmeur.

Nous commençons par étudier les traitements nécessaires afin de réaliser une visualisation (section 2), ainsi que les tâches qui doivent être supportées pour connecter une succession cohérente de visualisations (section 3). Nous introduisons ensuite l'architecture en services web permettant de réaliser les traitements et les rendre interactifs (section 4). Nous proposons une méthode de représentation et d'évaluation du parcours utilisateur dans ces visualisations interactives (section 5). Pour finir, nous discutons cette architecture et citons quelques perspectives possibles à donner à ces travaux (section 6).

2. Modèles de référence de traitement des données

D'un point de vue technique, un système visuel interactif transforme des données en images, qui sont des points de vue sur les données avec lesquels l'utilisateur peut interagir. Ces images sont le résultat de différents traitements qui suivent une longue chaîne qui commence dès la lecture d'un fichier de données ou l'interrogation d'une base de données, se poursuit par une transformation en structure de données interne (tableau, liste, graphe, etc.) qui sera ensuite disposée dans un espace et enfin associée à des variables graphiques. Cette chaîne aura un nom différent selon le domaine applicatif : en base de données, on parlera de QVT (*Query View Transform*), en intergiciels, de ETL (*Extraction Transformation Load*). Cette chaîne (appelée *"modèle de référence"* ou *"pipeline infovis"* (illustré figure 6) possède des étapes (section 2.1) et opérations (section 2.2) qui définissent l'espace de conception et d'interaction.

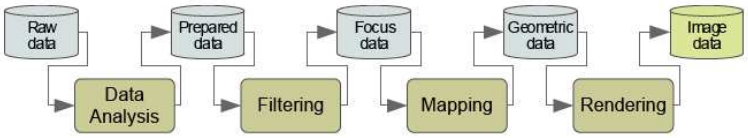

Figure 6. Un exemple d'étapes et d'opérations de traitement de données en visualisation d'information

2.1. Etapes de traitement des données

Les modèles de traitement de données (Haber, McNabb, 1990 ; E. H. Chi, 2000 ; Heer *et al.*, 2005 ; Dos Santos, Brodlie, 2004) sont des représentations graphiques des traitements et possèdent une *source* et une *finalité* communes. Concernant la *source*, il s'agit de données "brutes" [2], et la *finalité* un résultat visuel perceptible par l'utilisateur. Ces modèles diffèrent de par la granularité de leur découpage et leur terminologie, mais reposent sur la séparation des données de leur représentation graphique, ce qui est un motif classique de conception de type MVC (Krasner, Pope, 1988) (*Modèle-Vue-Contrôleur*). Il est cependant possible de regrouper ces étapes de traitements en trois parties complémentaires (figure 7) :

1. ***Abstraction de données :*** les données dérivées (Haber, McNabb, 1990), l'abstraction analytique (E. H. Chi, 2000), les données abstraites (Heer *et al.*, 2005) données préparées et focus (Dos Santos, Brodlie, 2004).

2. ***Abstraction visuelle :*** un objet visuel abstrait (Haber, McNabb, 1990), une abstraction visuelle (E. H. Chi, 2000), un analogue visuel (Heer *et al.*, 2005), données géométriques (Dos Santos, Brodlie, 2004).

2. En pratique il est difficile d'identifier des données brutes, car toute donnée résulte d'une vue partielle d'un dispositif de collecte (capteur, observation, etc.), qui possède une limite telle que la fréquence d'échantillonnage ou la catégorisation par l'humain.

3. ***Vue sur le rendu :*** une image affichable (Haber, McNabb, 1990), une vue (E. H. Chi, 2000), un affichage (Heer *et al.*, 2005), données d'image (Dos Santos, Brodlie, 2004).

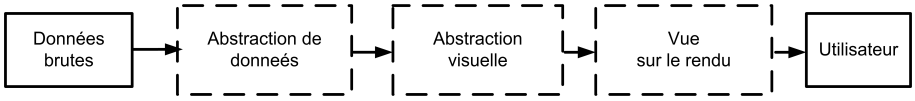

Figure 7. Etapes de traitement des données

Ces étapes sont assez génériques et peu formelles. Elles offrent malgré tout un cadre d'analyse et de comparaison pour la classification des techniques d'interaction visuelle (E. H. Chi, 2000), le positionnement d'un espace d'optimisation technique avec l'utilisation du GPU pour le rendu graphique (McDonnel, Elmqvist, 2009), ainsi que la répartition et l'équilibrage de traitements de données côté client ou côté serveur (Wood *et al.*, 1996).

2.2. Opérations sur les données

Les trois étapes précédemment énumérées constituent des états (intermédiaires) pour les données, au fil d'*opérations*. Ces opérations comportent des opérateurs de deux types (E. H.-h. Chi, Riedl, 1998 ; Stolte *et al.*, 2003) : opérateurs de *valeurs* (*value operator*) qui modifient les données, et les opérateurs de *vues* (*view operator*) qui modifient la projection des données (sans modifier les données sources). Un troisième type d'opérateur de *transformation de vue* est identifié (E. H. Chi, 2000) qui permet la manipulation des rendus par changement de point de vue (rotation, zoom, etc.).

Nous identifions trois catégories d'opérateurs communes (figure 8), associés respectivement aux étapes de traitement précédentes :

1. ***Transformation de données :*** enrichissement de données (Haber, McNabb, 1990), transformation des données (E. H. Chi, 2000), filtrage (Heer *et al.*, 2005).

2. ***Transformation d'association visuelle :*** cartographie visuelle (Haber, McNabb, 1990), transformation visuelle (E. H. Chi, 2000), filtrage (Heer *et al.*, 2005).

3. ***Transformation de vue :*** rendu (Haber, McNabb, 1990), transformation de cartographie visuelle (E. H. Chi, 2000), rendu (interactif) (Heer *et al.*, 2005).

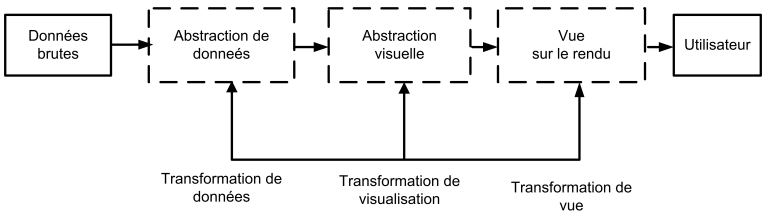

Figure 8. Opérateurs de traitement des données

Ces trois types d'opérateurs possèdent une forme de dépendance (E. H.-h. Chi, Riedl, 1998) ce qui augmente la difficulté d'identification de leur périmètre d'action. Par exemple, un opérateur de vue, qui filtre des éléments dans l'espace visuel, sans pour autant changer les données à la source, rend ces éléments invisibles alors qu'ils sont présents dans le jeu de données. De même une transformation de point de vue telle que le zoom peut faire varier la taille d'un objet, sans changer ni sa représentation visuelle (à savoir son association aux variables graphiques), ni ses valeurs.

3. Tâches et stratégies utilisateur d'exploration visuelle

Si l'objectif de l'utilisateur est d'explorer visuellement un jeu de données, la technique élémentaire serait de parcourir systématiquement toutes les abstractions de données (valeurs possibles à sélectionner ou filtrer), ainsi que les abstractions visuelles (encodages, position, etc.) et enfin les vues sur le rendu (zoom, rotation, etc.). Cette démarche n'est pas optimale car elle offre une combinaison importante de vues à explorer et ne pourrait être réalisable en un temps humain raisonnable. Une démarche structurée et méthodique doit donc être mise en place, afin de limiter toute forme d'opportunisme et de hasard dans la phase d'exploration visuelle. L'utilisation de connaissances préalables, telles que les tâches (section 3.1) et séquences de tâches (section 3.2) est une approche permettant de limiter et structurer l'espace d'exploration visuelle.

3.1. Tâches d'exploration visuelle

Un problème ou une tâche de "haut niveau" (comme l'identification de tendances, la recherche d'une corrélation, etc.) peut être difficile à appréhender par l'utilisateur. Il est donc nécessaire d'effectuer une décomposition en sous-problèmes (Dix *et al.*, 2004), réalisables facilement au moyen d'interactions ou de tâches élémentaires (Amar *et al.*, 2005) qui génèreront autant de visualisations de données. L'utilisateur devra ensuite rassembler ou coordonner les résultats afin de résoudre la tâche initiale.

À l'opposé, les techniques d'interaction de bas niveau, comme les BVI (*Basic Visualization Interaction*) (Chuah, Roth, 1996) sont une catégorisation hiérarchique en trois classes : les opérations graphiques (*graphical operations*), afin de changer l'apparence de la visualisation, les opérations sur les données (*data operations*), afin de manipuler les données, et enfin les opérations ensemblistes (*set operations*), pour créer et manipuler des objets.

Shneiderman (1996) propose une collection intermédiaire et générique de sept types de tâches de visualisation :

1. **Vue globale (*Overview*) :** *obtenir une vue globale de la collection complète*. La vue globale peut être une abstraction sur les données, qui ne modifie pas les données mais qui extrait une composante (structure, relation, dimension, etc.) et offre un cadre de réflexion propice à la prise de décision.

2. **Zoom (*Zoom*) :** *zoomer sur un objet intéressant afin d'obtenir plus de détails.*

3. **Filtrer (*Filter*) :** *filtrer les éléments non intéressants.* Notamment au moyen de widgets interactifs, tels que les requêtes dynamiques (Ahlberg, Shneiderman, 1994).

4. **Détails à la demande (*Details-on-demand*) :** *sélectionner un élément ou un groupe d'éléments et obtenir les détails quand il y en a besoin.*

5. **Mise en relation (*Relate*) :** *visualiser les relations entre les éléments.* Par exemple au moyen de juxtaposition de vues différentes sur les données, et une coordination entre les vues.

6. **Historique (*History*) :** *garder un historique des actions afin d'autoriser et corriger les erreurs (*undo*), rejouer et raffinement progressif.* Par exemple le système GRASPARC (Brodlie *et al.*, 1993) qui construit un arbre d'historique au fil des différentes interactions.

7. **Extraire (*Extract*) :** *permettre l'extraction de sous-collections et des paramètres des requêtes.*

Il existe de nombreux autres types de tâches, mais qui possèdent une terminologie propre, ce qui limite leur comparaison (Yi *et al.*, 2007 ; Munzner, 2009). Des tâches plus spécifiques aux structures de données peuvent être recensées comme pour les graphes (Lee *et al.*, 2006). Dès lors, ces tâches spécifiques ne sont pas généralisables, et donc pas extensibles aux autres structures de données (alors que les tâches génériques (Amar *et al.*, 2005 ; Shneiderman, 1996) peuvent, elles, s'appliquer à tout type de données).

3.2. Stratégies d'exploration visuelle

Les stratégies de navigation permettent d'agencer les tâches de manière cohérente, soit parce qu'elles sont dépendantes entre elles, soit parce qu'il est nécessaire de fournir à l'utilisateur une suite logique d'un point de vue cognitif. L'utilisateur doit pour sa part être attentif et essayer d'extraire des motifs au travers de variables graphiques telles que les regroupements, la taille ou les couleurs. À noter que les utilisateurs experts, qui possèdent déjà une bonne maîtrise des abstractions de données et visuelles, peuvent explorer plus rapidement l'espace et sauter des étapes.

La stratégie de navigation la plus souvent citée est le *visualization-information-seeking mantra* de (Shneiderman, 1996), issu d'années d'expériences en développement d'interfaces. Ce mantra propose une stratégie en trois étapes : *"Overview first, zoom and filter, then details-on-demand"*, qui reprend une partie des tâches énumérées dans la partie précédente. Démarrer par une vue globale n'est cependant pas le point de départ universel. Par exemple, dans le cas d'un graphe (Lee *et al.*, 2006) il peut être pertinent de réaliser une démarche liée à la topologie du graphe, comme commencer à partir d'un nœud et ensuite d'explorer ses voisins. La connaissance sera ainsi construite par associativité. L'analyse visuelle de données demande une démarche particulière. (Keim *et al.*, 2006) ont revu le mantra de Shneiderman en *"Analyse First -*

Show the Important - Zoom, Filter and Analyse Further - Details on Demand" qui est plus approprié à l'analyse visuelle de données.

Les stratégies de navigation visuelle peuvent être également locales à une vue. Par exemple l'intégration de *scented widgets* (Willett *et al.*, 2007) ou la prévisualisation de résultats (requêtes dynamiques et progressives), permettent de donner un aperçu de l'étape suivante, tel que la quantité de résultats attendus. L'animation est également une possibilité d'exploration afin d'éviter une série d'interactions répétitives (au détriment de l'interactivité, mais qui permet à l'utilisateur de se concentrer sur la visualisation). GAPMINDER (figure 3) permet de visualiser automatiquement l'évolution des données graphiques au fil des années. Une approche mixte est l'interface GOOGLE INSIGHTS FOR SEARCH (en bas à droite sur la figure 5) qui permet la lecture d'une vue seulement (la carte choroplèthe) ce qui laisse le reste de l'interface interactif et permet à l'utilisateur d'obtenir des détails à la demande.

Les stratégies de navigation peuvent être incluses dans les systèmes dès leur design, au moyen de motifs de conception (Chen, 2004). Mais un système peut rester ouvert à tout type de stratégie (Stolte *et al.*, 2002) *Often the path of exploration is unpredictable, and thus analysts need to be able to rapidly change both what data they are viewing and how they are viewing that data.*. À noter que la résolution des systèmes et leur taille influencent grandement la perception et l'efficacité qu'en a l'utilisateur (Yost *et al.*, 2007). Par exemple, même si en théorie tout jeu de donnée est visible avec un nombre infini de pixels, la réalité des écrans matriciels actuels et leur résolution variable limitent les possibilités d'affichage. Les stratégies de navigation doivent donc tenir compte de ces paramètres externes, ainsi que de paramètres liés à l'utilisateur (Tory, Möller, 2004) : possibilité de faire varier la résolution de l'écran, préférences globales, environnement physique, contexte social et culturel, etc.

4. Implémentation en services web interactifs de visualisations

Nous proposons une architecture logicielle permettant de reproduire le modèle de traitement de données (section 2), en une composition de services web (section 4.1). Les services web sont la transformation de programmes ordinaires en serveurs web autonomes et exécutés à distance, et peuvent être appliqués à la visualisation (section 4.2). Ces services peuvent être composés à d'autres services (section 4.3) afin de créer des visualisations qui seront rendues interactives une fois couplées à un programme interactif (section 4.4). L'interaction avec ces différents services permet la réalisation de tâches d'exploration visuelle précédemment identifiées (section 3), ainsi que le partage à l'échelle du web.

4.1. Services web de visualisation

Les services web permettent la mise en œuvre d'une architecture distribuée faiblement couplée. Ils sont définis par le W3C (Schlimmer, 2002) comme *a software application identified by a URI, whose interfaces and binding are capable of being de-*

fined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols et un client *a software that makes use of a Web Service, acting as its 'user' or 'customer'*. Les services web sont donc des programmes possédant une interface publique et standardisée, accessible via un réseau. Par conséquent, la localisation, l'environnement d'exécution, le langage d'implémentation et la méthodologie de développement ne sont pas contraignants. Ils permettent d'éliminer les problèmes d'interopérabilité au niveau matériel et plateforme d'exécution, et peuvent être utilisés par plusieurs applications.

Nous choisissons une approche de services web de type REST [3] (*REepresentational State Transfer*), afin de rester en phase avec Heer *et al.* (2008) qui indique que la collaboration sociale à l'échelle d'internet est facilitée par l'échange de simples URL. Cette approche permet à tout navigateur web d'interroger les services et permet de naviguer d'un service à un autre. Ce type de services web a été introduit en 2000 par R.T. FIelding (2008) pour qui *REST-based architectures communicate primarily through the transfer of representations of resources*. Il reprend le style de l'architecture *World Wide Web* et intègre également les caractéristiques suivantes :

– **Sans états :** c'est-à-dire que les paramètres passés au service doivent être suffisants pour exécuter le service sans que le serveur n'ait eu besoin au préalable de stocker des informations.

– **URI unique :** afin d'identifier les ressources sans ambiguïté. L'URL est une URI qui permet d'accéder à une ressource et composée d'un hôte, d'une chaîne de requête (*query string*).

– **Opérations HTTP :** permettent d'interroger de manière uniforme les services : `GET`, `POST`, `PUT`, et `DELETE`.

À noter que l'opération (dit *verbe*) `GET` est souvent la plus utilisée et implémentée par les serveurs web, car elle permet la transmission de paramètres dans l'URL. `POST` est également utilisée mais impose la transmission de paramètres dans le *corps* du message HTTP, ce qui ne permet pas de communiquer l'état dans l'URL (`POST` est principalement utilisé pour les services de type SOAP). Enfin les opérations `PUT` et `DELETE` ne sont pas toujours supportées par les serveurs web.

4.2. Création de services web de visualisation

La phase *technique* de création des services web repose sur une approche de création d'*adaptateurs* (*wrappers*) afin d'uniformiser les entrées et sorties de programmes

3. Cette approche est simple et largement utilisée, car elle consiste à permettre la publication et l'interrogation des services au-dessus de HTTP et au moyen d'une URL classique. Une alternative pourrait être SOAP (*Simple Object Access Protocol*), recommandée par le *Web Services Architecture* (WSA) qui est utilisée dans certains systèmes de distribution de visualisation (ADVISE (Wood *et al.*, 2008)) mais complexe à mettre en œuvre. Le système XWEB propose un nouveau protocole pour communiquer les interactions (Olsen *et al.*, 2000) (autre que HTTP), mais qui ne permet pas d'être interopérable à l'échelle du web.

existants [4] au moyen d'une API (Scuturici, 2009) [5] (*Application Programming Interface*, interface de programmation applicative). L'API (Scuturici, 2009) rend donc les interfaces des programmes uniformes et disponibles sur un réseau avec une IP et un port réseau. Un programmeur pourra ainsi adapter un script, une fonction, une source de données, etc. en un service web, avec de simples notions élémentaires de Java ou de C++. Ce type d'approche (*wrapping*) existe déjà pour des bibliothèques et plateformes de visualisation VTK (Gao, Chen, 2007), IRIS EXPLORER (Walton, 2004) et ADVISE (Wood *et al.*, 2008). Notre approche se différencie par sa simplicité et sa généricité à tout programme exécutable et tout type de source de données (l'API (Scuturici, 2009) a une implémentation en Java qui fonctionne sous WINDOWS, LINUX et MACOS).

4.3. Composition de services en flots de visualisation

Afin de reproduire les traitements (que nous appellerons *flots* par la suite) complets de traitement de données, il est nécessaire de proposer un mécanisme de composition des services en flots de visualisation. Le principe que nous utilisons est similaire aux *pipes* Unix [6], qui permettent de faire communiquer plusieurs programmes au moyen d'une interface commune. Les détails techniques des communications sont décrits dans l'API (Scuturici, 2009) [7], nous allons nous intéresser à la méthodologie de composition par l'utilisateur, afin de rendre notre architecture flexible et fortement paramétrable.

Composer les services peut être complexe car il s'agit d'intégrer de nombreux paramètres de correspondance entre les services (paramètres d'entrée et de sortie) tout en explorant l'espace de conception (choix des services). Nous avons donc proposé une interface de programmation visuelle MASHVIZ (Vuillemot, Rumpler, 2009a) (figure 9) qui permet aux utilisateurs de créer des flots en manipulant des éléments graphiques (symboles, agencement dans l'espace, etc.) qui représentent les services et leurs connexions. L'utilisation de ce type d'interface est courante dans des domaines très variés où les données sont sous forme de flux (Johnston *et al.*, 2004). C'est une approche mixte de programmation qui offre un niveau d'abstraction de composants qui permet à la fois d'impliquer des experts et des utilisateurs d'un domaine (*power*

4. À condition qu'il soit possible de transmettre au programme les paramètres d'entrées automatiquement, sans nécessiter l'usage d'une interface graphique. Certains logiciels (comme GIMP (Program, 2010) ou TULIP (Auber, 2003)) proposent une approche mixte avec une interface graphique utilisateur, et une interface de script. De même pour la sortie du programme.

5. Cette API a été développée dans le cadre du projet "dataspace" au LIRIS http://ds.liris.cnrs.fr/ et vise à fournir une vue intégrée et dynamique sur un espace de données issues de capteurs, bases de données, fichiers, etc. sous forme de services web de type REST.

6. http://en.wikipedia.org/wiki/Pipeline_(Unix)

7. Les communications entre services se réalisent au moyen d'URL. Par exemple l'URL suivante permet de connecter l'entrée d'un service de construction de graphes à une sortie source de données (nombres aléatoires) : http://127.0.0.1:1713/GraphBuilder/input/Data/connection/add?URL=http://127.0.0.1:8015/RandomNumber/output/number

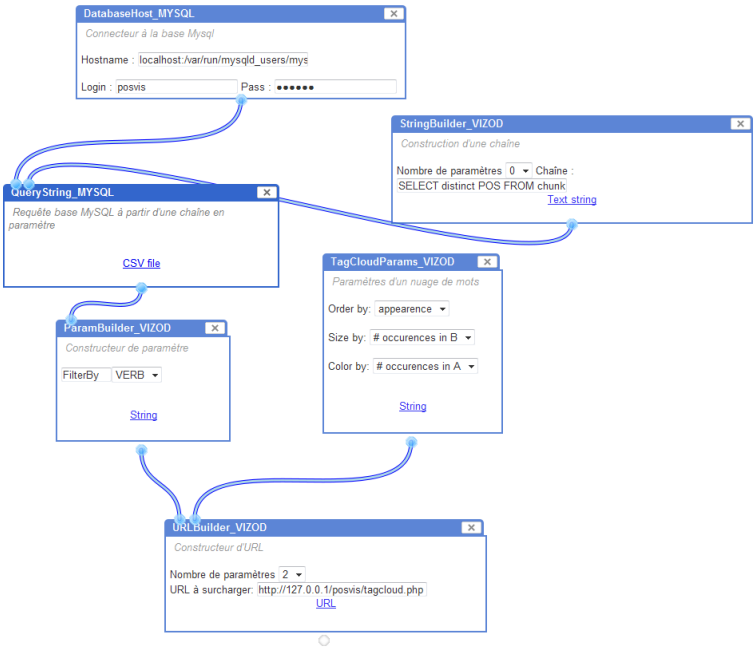

Figure 9. Flot de visualisation résultant en un nuage de mot-clés. Le flot débute en service de requête dans une base de données (en haut), et termine par un service de construction d'URL (en bas) pour accéder au nuage de mots-clés

user). L'objectif d'une telle interface est de permettre à un utilisateur de comprendre et de manipuler du code à un niveau intuitif, et lui fournir un ensemble d'interactions possibles, qui permettent également d'éviter des erreurs (de bas niveau, comme éviter le choix de paramètres incompatibles entre deux services).

MASHVIZ s'inscrit dans un écosystème plus large de création et de partage des services et des flots. Pour cela nous avons identifié trois types d'acteurs (Vuillemot, Rumpler, 2009b) : *programmeurs* (qui programment une machine en fonction des besoins d'utilisateurs ou de designers), *designers* (qui manipulent des briques déjà produites et qui s'intéressent à la structure globale) et *utilisateurs* (qui utilisent un système afin de réaliser une tâche).

L'implémentation de MASHVIZ (figure 9) est orientée web (et ne dépend donc pas d'un système d'exploitation) et coordonne des services qui sont exécutés à distance. Elle reprend le *look and feel* de Yahoo! Pipes (Pipes, 2010) pour l'apparence générale, l'adaptation des services en boîtes, et la connexion élastique par *tuyaux* en courbes de Bézier. L'utilisateur possède donc un plan de travail dans lequel des services peuvent être glissés/déposés depuis la *base de service*, qui a été divisée en catégories d'étapes de traitement des données (Vuillemot *et al.*, 2009). Chaque service déposé peut être personnalisé dans la boîte qui le représente. Cette boîte est publiée par le service lui-

même et est affichée sous forme de sous-page web dans la boîte. Ensuite, l'utilisateur peut connecter/déconnecter les entrées et sorties des services afin de créer un flot. Il n'est pas possible pour l'utilisateur de relier des boîtes incompatibles.

Partager la composition de services qui constitue un flot permet une analyse d'usage plus fine que les approches classiques de partage de visualisation (telles que ManyEyes), car il ne s'agit plus de partager le rendu (image) ou l'abstraction de données (comme un jeu de données) mais *toutes* les étapes qui constituent le résultat (dans la mesure où la granularité des services est assez fine). Les données d'usage sont visibles pour le fournisseur des services, et des tâches supplémentaires de surveillance et de débogage d'exécution peuvent être envisagées et affichées dans MASHVIZ [8].

4.4. Interaction avec les services de visualisation

Les services et flots de visualisations précédemment créés sont exécutés une fois qu'une application envoie une requête via le protocole HTTP au dernier service du flot, qui ensuite propage la requête aux autres services. Cette requête est identique à l'appel d'une page web par un navigateur et doit comporter l'URL du service ou du flot avec les paramètres dans celle-ci ou dans l'en-tête HTTP. En retour, le service renvoie une chaîne de caractères composée d'un code (Fielding *et al.*, 1999) qui permet (entre autres) d'indiquer si la ressource (programme adapté par le service) est disponible et le résultat transféré (`200 OK`), si les droits d'accès ne sont pas suffisants (`403 Forbidden`) ou si la ressource n'est pas trouvée (`404 Not Found`). Les paramètres (transmis dans l'URL de la requête) et le format de retour des données (transmis dans le corps du message de retour) sont définis par l'API (Scuturici, 2009).

4.5. Exemples

4.5.1. Nuage de mots-clés

Nous présentons la conception d'une visualisation d'information classique : le nuage de mots-clés. Il s'agit de représenter de manière compacte une sélection de mots d'un texte, avec une couleur, taille et position encodant des données telles que la fréquence, l'ordre d'apparition dans le texte ou le type de mot (entité nommée, caractéristiques grammaticales).

Le résultat est illustré figure 9 (la création et la composition de services sont détaillés dans (Vuillemot, 2010)). L'URL [9] doit ensuite être intégrée dans une application interactive (par exemple une application JAVA SWING ou un navigateur web), et les

8. Mais cela est encore considéré comme un problème complexe pour les interfaces de programmation visuelle (Johnston *et al.*, 2004).

9. L'URL d'accès au service est de la forme suivante : http://HOST/posvis/tagcloud.php?chapter=1&collection=1&startSelectionA=0&endSelectionA=4892&startSelectionB=4893&endSelectionB=&orderBy=appearence&sizeBy=frequencyInB&colorBy=frequencyInA&minSize=50&maxSize=500&limitMinSize=50

widgets de l'application joueront le rôle de contrôleur et passeront les variables en paramètre de l'URL du service. Ainsi il sera possible d'effectuer des actions de filtrage, sélection, changement de modèle de données ou de mapping visuel (Riche *et al.*, 2010).

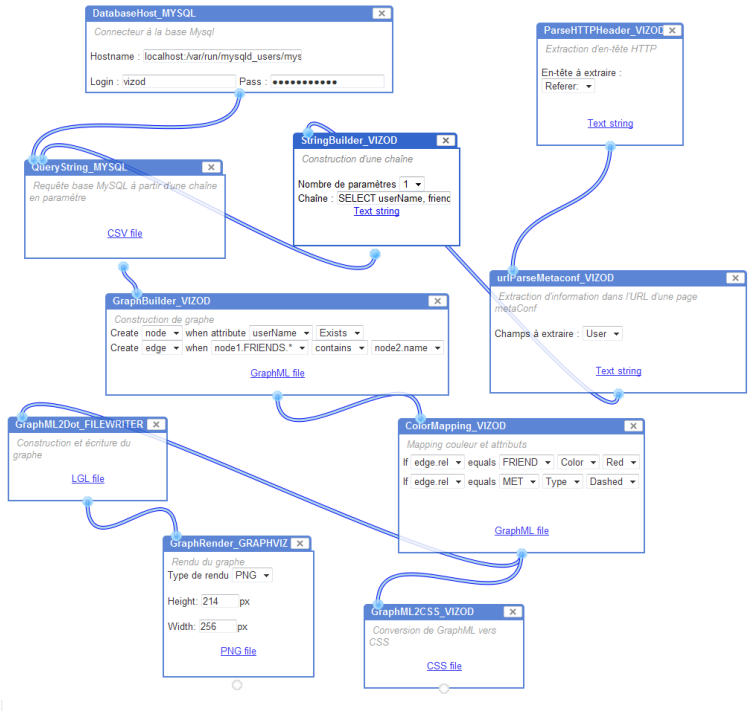

Figure 10. Flot de visualisation d'un graphe de type nœud/lien. Le flot possède deux sorties : l'image du graphe (image de type PNG) et l'encodage visuel (feuille de style CSS)

4.5.2. Graphe nœud-liens

La visualisation de graphes est également une visualisation classique, dont deux des principaux challenges sont 1) déterminer la bonne disposition des sommets et des arêtes dans l'espace et 2) utiliser la bonne stratégie d'exploration.

La création d'un graphe de type réseau social est illustrée figure 10 (la création et la composition de services sont détaillées dans (Vuillemot, 2010). Le résultat est une image statique (figure 11, gauche). Il est possible de changer le point de vue sur celle-ci avec un environnement interactif avancé tel que Google Earth. Le changement de point de vue permettra également de détailler le graphe au fil du zoom sur celui-ci (figure 11, droite).

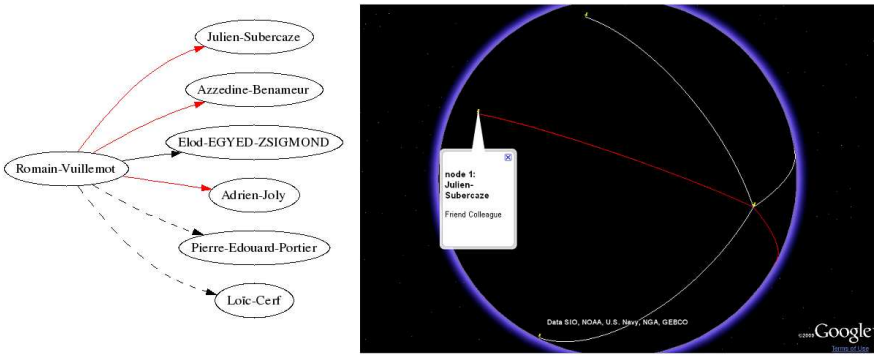

Figure 11. Résultat de l'image issue du flot de visualisation d'un graphe (gauche) et interaction dans Google Earth (droite)

5. Evaluation des interfaces visuelles

L'étude de l'usage des flots de visualisation, l'interaction de l'utilisateur avec ceux-ci et le passage d'un flot à un autre constituent la base de notre approche d'évaluation. Nous allons présenter deux types d'évaluations à base de métriques quantitatives d'exécution des flots (section 5.1), et d'analyse du parcours de l'utilisateur au fil des flots (section 5.2).

5.1. Métriques quantitatives

Les services web et l'application cliente offrent de nombreuses métriques *quantitatives* de suivi et d'analyse de l'usage de l'utilisateur (exemple de métriques tableau 1). Toute application interactive basée sur l'implémentation d'une architecture en services web (à savoir pas uniquement celle que nous venons d'introduire) peut donc être systématiquement instrumentalisée [10] et évaluée selon ces métriques. Les données issues de ces métriques sont générées sur les machines hébergeant les services web et les profils applicatifs, ou par un proxy intermédiaire (Hong *et al.*, 2001) et sont stockées dans des logs. Cette automatisation permet notamment de faciliter l'évaluation à long terme d'interfaces visuelles (Shneiderman, Plaisant, 2006) [11] mais ne permet pas d'expliquer la cause d'une action ou de détecter tout facteur externe (discussion, annotation sur un morceau de papier, etc.) à l'environnement informatique.

Des métriques spécifiques propres aux flots de visualisation peuvent également être introduites : nombre et fréquence des services utilisés, paramétrage invariant (comme le mapping visuel) entre les flots, etc. Enfin des métriques propres aux ar-

10. Dans le cadre d'un respect de la vie privée, ce qui implique un accord explicite ou une anonymisation totale de l'utilisateur.

11. Toutes les métriques ne sont pas à utiliser, le choix de leur pertinence est laissé aux évaluateurs en fonction de la tâche, et de la quantité de données d'usage générées.

Tableau 1. Liste de métriques quantitatives d'évaluation

Module	Indicateur	Unité
Services et flots	Temps moyens d'exécution	ms
	Latence du réseau	ms
	Débit du réseau	Kb/s
	Jitter (variabilité de la latence)	ms
Application cliente	Réactivité	ms
	Rétroactivité	ms
	Framerate	$images/s$

chitectures découplées à haute performance et distribuées [12], permettraient d'identifier les éventuels goulots d'étranglement (*bottlenecks*).

5.2. Identification de l'activité utilisateur

Au-delà de mesures quantitatives, il est également possible d'identifier les tâches et stratégies utilisateur. Cela nécessite une analyse du comportement sur une session complète, et non plus sur un ou plusieurs services pris indépendamment.

Pour illustrer notre démarche, nous reprenons l'interface visuelle utilisée pour explorer un graphe généré à partir de la composition de services (section 4.5.2). Pour rappel, ce graphe résulte en une image qu'il est parfois difficile d'explorer de par sa taille ou le grand nombre d'attributs des données. Une approche possible pour limiter la surcharge visuelle et le nombre d'interactions, est de montrer une vue globale pour ensuite prendre une décision et affiner le parcours jusqu'au niveau minimal de zoom et maximal de détails (Shneiderman *et al.*, 2009). Une contrainte majeure est de garder une certaine consistance dans les différentes résolutions, nous avons pour cela gardé la même disposition quelle que soit la résolution.

Nous avons choisi une navigation zoomable, où la disposition du graphe reste constante, sur laquelle l'utilisateur pourra donc zoomer en avant et en arrière comme une carte normale. Le niveau de détail variera, soit automatiquement selon le niveau de zoom, soit par filtrage manuel (via des listes latérales). L'interface cliente que nous avons utilisée est Google Earth (dans lequel nous avons supprimé toute référence géographique), et nous y faisons varier la résolution du graphe en fonction de l'altitude de l'utilisateur (figure 11, droite).

La phase d'évaluation se base sur les appels d'URL générés par l'application Google Earth, et la construction d'un diagramme d'états qui représente l'activité de l'utilisateur. Cette étape consiste à analyser des logs structurés du serveur qui héberge les flots et qui nous donnent des informations d'accès aux services (identifiant utilisateur, identifiant application, timestamp, etc.). Contrairement à l'analyse classique de

12. http://en.wikipedia.org/wiki/Application_Response_Measurement

log (comme par exemple dans un moteur de recherche (Adar *et al.*, 2008)), le vocabulaire d'appels est limité et il nous est donc possible de connaître chacun des états sans ambiguité.

La figure 12 montre une série de diagrammes automatiquement générés et mis à jour au fur et à mesure de l'activité d'exploration visuelle d'un graphe, à partir des logs. Les états des diagrammes sont représentés par des carrés aux bords arrondis (qui correspondent à un flot de visualisation, résultant chacun en un rendu du graphe), et les transitions d'un état à l'autre par une flèche. Le nom des carrés est le nom associé aux vues sur les données (vue globale, zoom, détails) qui sont des annotations incluses lors de la création des flots dans MASHVIZ. L'étiquette des flèches est le nombre de transitions entre ces vues. A gauche, sont représentés les diagrammes qui synthétisent les activités de plusieurs utilisateurs, au cours de plusieurs sessions d'utilisation. A droite, ces diagrammes sont agrégés en un seul afin de faire émerger le motif général de la navigation.

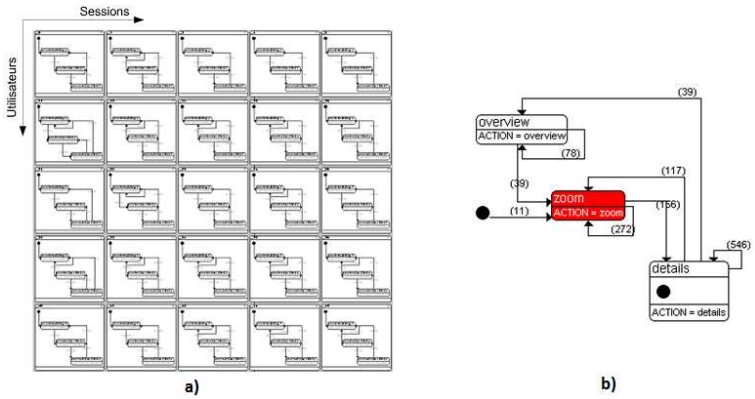

Figure 12. Diagrammes d'analyse du parcours d'utilisateur: a) Vue détaillée des sessions et des utilisateurs, b) vue agrégée

Ces diagrammes permettent dans un premier temps de comparer les différences de comportement des utilisateurs. Ensuite, l'évaluation à proprement parler consiste à effectuer la correspondance entre les stratégies d'exploration attendues (section 3) et l'observation synthétisée dans ces diagrammes. Dans ce sens, un résultat que nous pouvons mettre en avant est que les utilisateurs commencent l'application (rond noir foncé) à un état de zoom, alors qu'ils devraient commencer par la vue globale (selon (Shneiderman *et al.*, 2009)). Une deuxième remarque est que beaucoup d'utilisateurs repassent à la vue globale directement sans passer par l'état de zoom, cela indique au designer qu'il serait utile de mettre en place un moyen rapide de passer entre ces deux étapes (sans passer par le zoom intermédiaire).

Le diagramme reste cependant une vue restreinte sur l'activité réelle, car 1) l'observation est du côté serveur, et 2) chaque étape est étiquetée selon des sous-tâches principales par le programmeur (de flots de visualisation). Le rôle de ces étapes est

souvent très contextuel, il peut avoir un sens différent selon l'étape précédente. Enfin, la granularité des appels n'est pas assez fine pour avoir une bonne connaissance de l'activité réelle de l'utilisateur sur une vue. Par exemple il peut effectuer des rotations au niveau du zoom, ou tout type d'interaction qui n'envoie pas d'appel au serveur.

6. Discussion et perspectives

L'usage des services web pour la création de visualisations interactives a l'avantage de se baser sur une architecture et des standards web existants, tout en l'élargissant à la création de visualisations interactives. Nous avons certes introduit nos propres outils d'adaptation en services (Scuturici, 2009), de composition (Vuillemot, Rumpler, 2009a) et d'évaluation en diagrammes d'états, mais d'autres outils auraient pu être utilisés. Par exemple, l'évaluation de l'usage des flots peut être assimilée à un comportement utilisateur classique sur le web et il existe de nombreux outils permettant leur suivi et analyse (Google Analytics, etc.).

L'analyse de l'activité utilisateur étant plus transparente, de nouvelles perspectives de partages de vues de navigation (Shrinivasan, Wijk, 2008) s'ouvrent. En effet, non seulement toute visualisation peut être communiquée avec les traitements qui ont permis de la produire, mais également les interactions. Cela permettrait peut-être une meilleure compréhension du raisonnement de l'utilisateur qui l'a produite. Cela permettrait également une personnalisation et une reproduction des résultats avec un autre jeu de données. Un autre point à explorer serait d'automatiquement extraire une sémantique de la vue : vue globale ? vue détaillée ? vue intermédiaire de transition animée ? Dans le cas de l'exploration de graphe (section 4.5.2) elles ont été préalablement annotées ce qui permet de connaître leur sémantique lors de l'évaluation. Enfin, la granularité des services, à savoir le nombre d'étapes de traitement de données qu'ils intègrent, est un équilibre complexe à gérer. Ceux-ci peuvent aller du simple calcul arithmétique à la mise à jour de données dans des bases de données. Trop de services rendent l'interface de composition certes hautement personnalisable, mais trop complexe à manipuler. Réciproquement, trop peu de services ne permettraient pas assez de personnalisation, mais simplifieraient leur composition.

Enfin, nous pensons que de nouvelles méthodes de coopération peuvent émerger afin de permettre une analyse sociale de données (Heer *et al.*, 2008). MASHVIZ possède un annuaire de visualisations qui répertorie tous les flots de visualisation créés et les services existants, dans le but de faciliter leur réutilisation. Au-delà de la validation, l'analyse simplement *statistique* de données d'usage de services et de flots permettrait une exploration systématique de l'espace de design. Par exemple si certains paramètres ou combinaison de services n'ont jamais été testés, alors MASHVIZ serait en mesure d'indiquer cela. A l'inverse, un système de recommandation de flots pourrait suggérer aux utilisateurs, suite au parcours dans n précédent flots, les prochains flots les plus pertinents. Un utilisateur qui a créé une visualisation d'un graphe, peut se voir suggérer différentes alternatives de design : changement de mapping visuel, stratégie d'explorations en fonction des caractéristiques des données, etc. Soit

en fonction de ce qui a déjà été fait, soit afin d'explorer de nouvelles alternatives. Autrement dit, il s'agirait de passer d'un cadre d'analyse et d'évaluation, à un cadre génératif de nouveaux parcours visuels.

Bibliographie

Adar E., Teevan J., Dumais S. T. (2008). Large scale analysis of web revisitation patterns. In *Chi '08: Proceeding of the twenty-sixth annual sigchi conference on human factors in computing systems*, p. 1197–1206. New York, NY, USA, ACM.

Ahlberg C., Shneiderman B. (1994). Visual information seeking using the filmfinder. In *Chi '94: Conference companion on human factors in computing systems*, p. 433–434. New York, NY, USA, ACM.

Amar R., Eagan J., Stasko J. (2005). Low-level components of analytic activity in information visualization. In *Ieee symposium on information visualization, 2005. infovis 2005*, p. 111–117.

Auber D. (2003). Tulip-a huge graph visualization framework. *Graph Drawing Software*.

Brodlie K., Poon A., Wright H., Brankin L., Banecki G., Gay A. (1993). Grasparc: a problem solving environment integrating computation and visualization. In *Vis '93: Proceedings of the 4th conference on visualization '93*, p. 102–109. Washington, DC, USA, IEEE Computer Society.

Card S., Mackinlay J., Shneiderman B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

Chen H. (2004). Towards design patterns for dynamic analytical data visualization. In *Proceedings of spie visualization and data analysis*, p. 75–86.

Chi E. H. (2000). A taxonomy of visualization techniques using the data state reference model. In *INFOVIS*, p. 69-76. citeseer.ist.psu.edu/chi00taxonomy.html

Chi E. H.-h., Riedl J. (1998). An operator interaction framework for visualization systems. In *Infovis '98: Proceedings of the 1998 ieee symposium on information visualization*, p. 63–70. Washington, DC, USA, IEEE Computer Society.

Chuah M. C., Roth S. F. (1996). On the semantics of interactive visualizations. In *Infovis '96: Proceedings of the 1996 ieee symposium on information visualization (infovis '96)*, p. 29. Washington, DC, USA, IEEE Computer Society.

Dix A., Finlay J., Abowd G. (2004). *Human-computer interaction*. Prentice hall.

Dos Santos S., Brodlie K. (2004). Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, vol. 28, n° 3, p. 311–325.

Fekete J.-D., Wijk J. J., Stasko J. T., North C. (2008). The value of information visualization. In A. Kerren, J. T. Stasko, J.-D. Fekete, C. North (Eds.), *Information visualization*, p. 1–18. Berlin, Heidelberg, Springer-Verlag. http://dx.doi.org/10.1007/978-3-540-70956-5_1

Fielding R., Gettys J., Mogul J., Berners-Lee T. (1999). *Hypertext transfer protocol–http/1.1*. RFC 2616, June.

Gao S., Chen D. (2007). Applying web service technology in Distributed Visualization. In *2007 international conference on machine learning and cybernetics*, vol. 7.

GapMinder. (2010). http://www.gapminder.org/.

Google Insights for Search. (2010). http://www.google.com/insights/search/.

Haber R., McNabb D. (1990). Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in scientific computing*, p. 74–93.

Heer J., Card S. K., Landay J. A. (2005). prefuse: a toolkit for interactive information visualization. In *Chi '05: Proceedings of the sigchi conference on human factors in computing systems*, p. 421–430. New York, NY, USA, ACM.

Heer J., Ham F., Carpendale S., Weaver C., Isenberg P. (2008). Creation and collaboration: Engaging new audiences for information visualization. , p. 92–133.

Hibbard B. (2004). The top five problems that motivated my work. *IEEE Comput. Graph. Appl.*, vol. 24, n° 6, p. 9–13.

Hong J., Heer J., Waterson S., Landay J. (2001). WebQuilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems (TOIS)*, vol. 19, n° 3, p. 263–285.

Johnston W. M., Hanna J. R. P., Millar R. J. (2004). Advances in dataflow programming languages. *ACM Comput. Surv.*, vol. 36, n° 1, p. 1–34.

Keim D. A., Mansmann F., Schneidewind J., Ziegler H. (2006). Challenges in visual data analysis. In *Iv '06: Proceedings of the conference on information visualization*, p. 9–16. Washington, DC, USA, IEEE Computer Society.

Krasner G., Pope S. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming*, vol. 1, n° 3, p. 26–49.

Lee B., Plaisant C., Parr C. S., Fekete J.-D., Henry N. (2006). Task taxonomy for graph visualization. In *Beliv '06: Proceedings of the 2006 avi workshop on beyond time and errors*, p. 1–5. New York, NY, USA, ACM.

McDonnel B., Elmqvist N. (2009). Towards utilizing gpus in information visualization: A model and implementation of image-space operations. *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, p. 1105-1112.

Munzner T. (2009). A nested process model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, n° 6, p. 921–928.

Norman D. (2002). *The design of everyday things*. Basic Books New York.

Olsen D. R., Jr., Jefferies S., Nielsen T., Moyes W., Fredrickson P. (2000). Cross-modal interaction using xweb. In *Uist '00: Proceedings of the 13th annual acm symposium on user interface software and technology*, p. 191–200. New York, NY, USA, ACM.

Pipes Y. (2010). http://pipes.yahoo.com/pipes/. http://developer.yahoo.com/yui/

Program G. I. M. (2010). http://www.gimp.org/. http://www.gimp.org/

Riche N., Lee B., Plaisant C. (2010). Understanding Interactive Legends: a Comparative Evaluation with Standard Widgets. In *Computer graphics forum*, vol. 29, p. 1193–1202.

Schlimmer J. (2002). Web services description requirements. *World Wide Web Consortium (W3C) working draft. Oct*, vol. 28.

Scuturici M. (2009, septembre). *Dataspace API*. Rapport technique. LIRIS. http://ds.liris.cnrs .fr/

Shneiderman B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Vl '96: Proceedings of the 1996 ieee symposium on visual languages*, p. 336. Washington, DC, USA, IEEE Computer Society.

Shneiderman B., Plaisant C. (2006). Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Beliv '06: Proceedings of the 2006 avi workshop on beyond time and errors*, p. 1–7. New York, NY, USA, ACM.

Shneiderman B., Plaisant C., Cohen M., Jacobs S. (2009). *Designing the user interface: Strategies for effective human-computer interaction*. USA, Addison-Wesley Publishing Company.

Shrinivasan Y. B., Wijk J. J. van. (2008). Supporting the analytical reasoning process in information visualization. In *Chi '08: Proceeding of the twenty-sixth annual sigchi conference on human factors in computing systems*, p. 1237–1246. New York, NY, USA, ACM.

Stasko J., G
"org C., Liu Z. (2008). Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, vol. 7, n° 2, p. 118–132.

Stolte C., Tang D., Hanrahan P. (2002). Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, vol. 8, n° 1, p. 1.

Stolte C., Tang D., Hanrahan P. (2003). Multiscale visualization using data cubes. *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, n° 2, p. 176–187.

Takatsuka M., Gahegan M. (2002). GeoVISTA Studio: a codeless visual programming environment for geoscientific data analysis and visualization* 1. *Computers & Geosciences*, vol. 28, n° 10, p. 1131–1144.

Thomas J. J., Cook K. A. (2005). *Illuminating the path: The research and development agenda for visual analytics*. National Visualization and Analytics Ctr. Paperback.

Tory M., Möller T. (2004). Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, n° 1, p. 72–84.

Vuillemot R. (2010). *Un cadre de conception pour la Visualisation d'Information Interactive*. Thèse de doctorat en informatique, INSA de Lyon.

Vuillemot R., Rumpler B. (2009a). Une interface de programmation visuelle pour la composition de services de visualisation d'information. In *Ihm'09 : Actes de la 21ème conférence francophone sur l'interaction homme-machine.*

Vuillemot R., Rumpler B. (2009b, October). A web-based interface to design information visualization. *MEDES 2009*. http://liris.cnrs.fr/publis/?id=4459

Vuillemot R., Rumpler B., Pinon J.-M. (2009, avril). Dissection of a Visualization On-Demand Server. In J. C. Joaquim Filipe (Ed.), *Enterprise information systems*, p. 348–360. Springer Berlin Heidelberg. http://liris.cnrs.fr/publis/?id=3851

Walton J. (2004). NAG's IRIS explorer. *Visualization handbook*, p. 633–654.

Wattenberg M., Kriss J., McKeon M. (2007). Manyeyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, n° 6, p. 1121–1128. (Member-Viegas, Fernanda B. and Member-van Ham, Frank)

Willett W., Heer J., Agrawala M. (2007). Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, n° 6, p. 1129–1136.

Wood J., Brodlie K., Seo J., Duke D., Walton J. (2008). A web services architecture for visualization. In *Proceedings of the ieee fourth international conference on escience, 2008.*, p. 1–7.

Wood J., Brodlie K., Wright H. (1996). Visualization over the world wide web and its application to environmental data. In *Vis '96: Proceedings of the 7th conference on visualization '96*, p. 81–ff. Los Alamitos, CA, USA, IEEE Computer Society Press.

Yi J. S., Kang Y. a., Stasko J., Jacko J. (2007). Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, n° 6, p. 1224–1231.

Yost B., Haciahmetoglu Y., North C. (2007). Beyond visual acuity: the perceptual scalability of information visualizations for large displays. In *Chi '07: Proceedings of the sigchi conference on human factors in computing systems*, p. 101–110. New York, NY, USA, ACM.